Info 1 - Bases pour débuter avec Python- *Corrigé*

On débute ici nos premiers codes en langage Python que l'on écrira en utilisant l'interface **Pyzo** que vous avez déjà utilisé en classe de seconde. Cette interface présente l'avantage d'être simple d'accès.

En cours d'année, on l'abandonnera pour se tourner vers le produit Microsoft Visual Studio Code qui propose des outils plus efficaces.

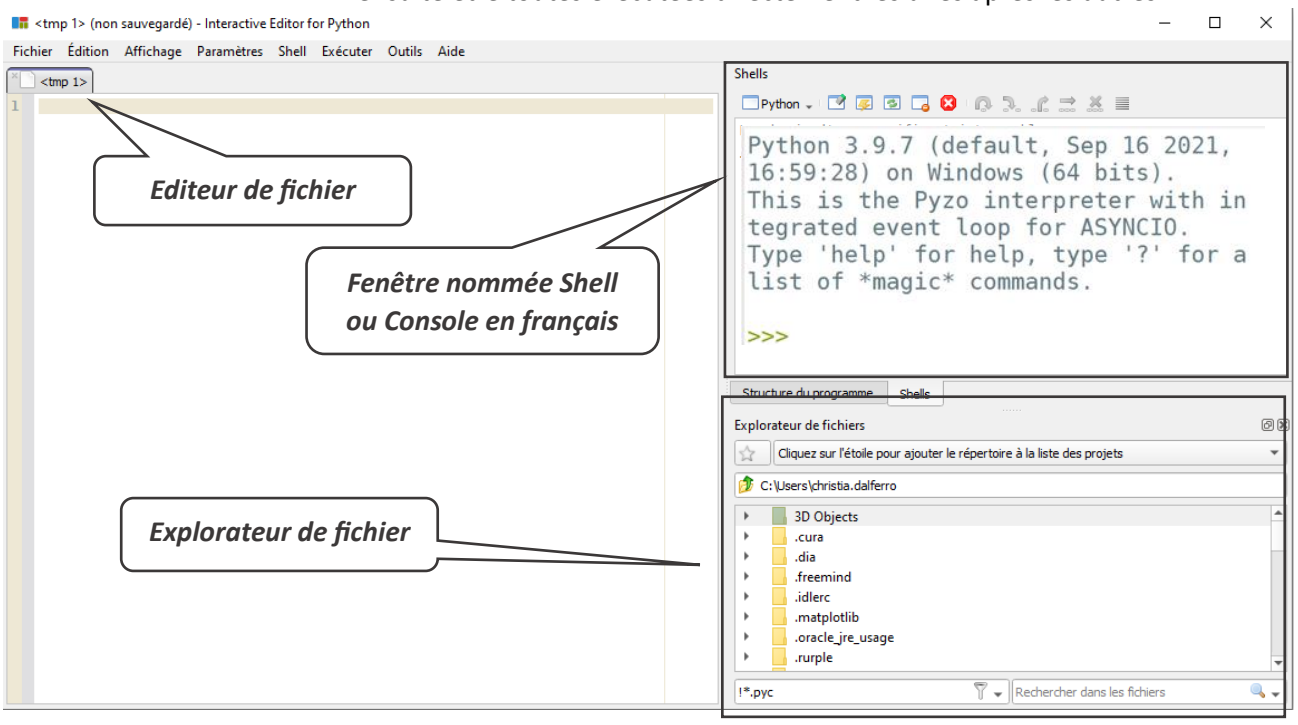
1- PRESENTATION DE L'I.D.E. PYZO

⇨ Lancer le logiciel ***Pyzo*** en ouvrant le lien nommé « ***Pyzo_sti2d-Raccourci*** » qui se trouve dans le dossier « ***Autres raccourcis*** / **Pyzo**» disponible sur le bureau de votre ordinateur.

Dossier « *Autres raccourcis* »

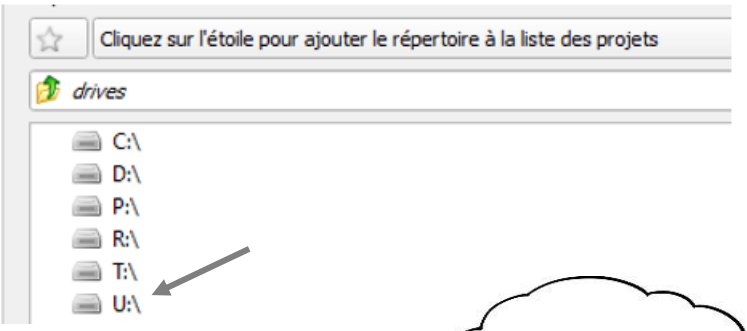
Ce logiciel nommé ***Pyzo*** est ce qu'on appelle un **environnement de développement intégré** (I.D.E). Il vous permet de développer plus facilement des codes informatiques en langage Python. *Pyzo* propose à l'ouverture 3 fenêtres :

- **La console** ou shells sur Pyzo, permet :
  - d'exécuter des instructions Python individuellement,
  - de connaître l'état des variables,
  - d'afficher des messages pour l'utilisateur.
- **L'explorateur de fichiers** permet de visualiser le contenu du répertoire de travail
- **L'éditeur** permet d'écrire des lignes de code python dans un fichier qui pourront ensuite être toutes exécutées directement les unes après les autres.



A l'ouverture de Pyzo, l'explorateur de fichier cible votre dossier dans *C:\Users* . Prenez l'habitude au départ de modifier ce dossier, **en pointant votre** zone de travail **sur le serveur** nommé

U:\ votreNom\Mes documents\...



2- PREMIERES LIGNES DE CODE PYTHON DANS LA CONSOLE

⇨ Écrire $5 - 90$ derrière les 3 chevrons `>>>` de la console et exécuter cette opération en tapant sur la touche *Entrée* du clavier.

⇨ Exécuter à présent l'opération $4 * 5$ ….

La console permet donc de réaliser des calculs facilement, un peu comme dans une calculatrice.

3- UTILISATION D'UNE PREMIERE VARIABLE NOMMEE x

On utilise à présent la notion de variable.

⇨ Exécuter toujours dans la console, la commande : `>>> x=10`

⇨ Exécuter ensuite la commande : `>>> x` …… normalement ***Python*** retourne la valeur de la variable x

⇨ Exécuter la commande : `>>> 4*x` , puis `>>> 4+x`
…….. ***Python*** renvoie le résultat du calcul avec x qui a une valeur de 10 .

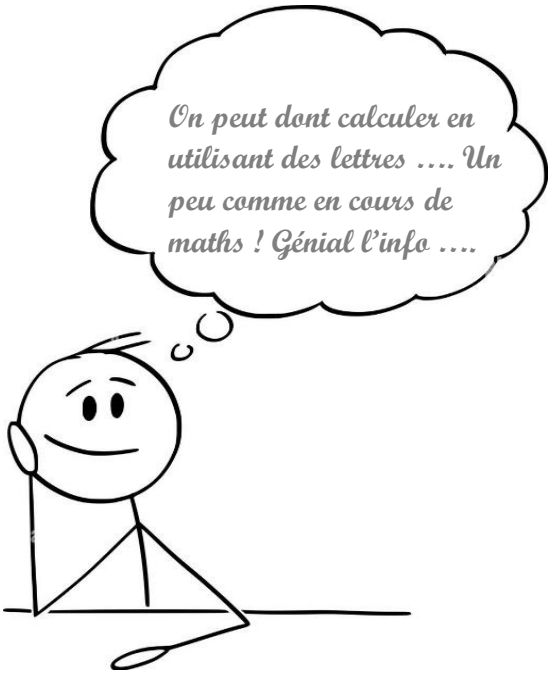
⇨ Changer la valeur de la variable x : `>>> x=-2023` et exécuter à nouveau les opérations `>>> 4*x` et `>>> 4+x` . …… ***Python*** refait les calculs en donnant à x **sa** nouvelle valeur qui est de -2023 .

4- UTILISATION DE DEUX VARIABLES NOMMEES a ET b

⇨ Exécuter les commandes `>>> a=11` , puis `>>> b=9`

⇨ Exécuter ensuite la commande : `>>> a+b`
…….. le résultat donné est-il correct ?

⇨ Exécuter ensuite la commande : `>>> (a-1)*(b+1)`
……. le résultat donné est-il correct ?



5- TYPES DES VARIABLES

Le type d'une variable correspond à la nature de celle-ci. Les quatre principaux types dont nous aurons besoin dans un premier temps sont **les entiers** (integer ou int), **les nombres décimaux** que nous appellerons floats, **les chaînes de caractères** (string ou str) et **les booléens** (true et false).

⇨ Exécuter les instructions suivantes dans le *shell* et analyser les résultas de chaque exécution ….

```
>>> a = "Bienvenue"
>>> type(a)
<class 'str'>
```

```
>>> b = a + ' en NSI, la promo '
>>> b
'Bienvenue en NSI, la promo '
```

```
>>> c = b + str(2023)
>>> c
'Bienvenue en NSI, la promo 2023'
```

```
>>> d = "L'info"+" "+'c"est'+" super"+"\nquand ça marche"
>>> d
'l\'info c"est super\nquand ça marche'
>>> print(d)
l'info c"est super
quand ça marche
```

```
>>> e = "Une règle de base : "+"\n"+"    ne t'énerve jamais\n"*10
>>> print(e)
Une règle de base :
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
    ne t'énerve jamais
```

```
>>> f = 5//2
>>> f
2
```

```
>>> f = 5%2
>>> f
1
```

```
>>> type(f)
<class 'int'>
```

```
>>> f == 2
False
```

```
>>> type(f == 2)
<class 'bool'>
```

```
>>> g = "L'info" == "trop facile"
>>> g
False
```

```
>>> h = "info" > "facile"
>>> h
True
```

```
>>> type(h)
<class 'bool'>
```

```
>>> x = 2.01
>>> type(x)
<class 'float'>
```

```
>>> x - 0.01
1.9999999999999998
```

….. ne devrait-on pas normalement trouver 2 car 2.01 – 0.01 = 2 ???

```
>>> x/0.01
200.99999999999997
```

```
>>> x * 0.01
0.0201
```

```
>>> y = input("Souhaites-tu avoir ton bac ? ")
>>> y
>>> type(y)
<class 'str'>
>>> if y == 'oui' or y == 'OUI' : print("On verra bien, à Dieu vat !")
>>> int(y)
ValueError: invalid literal for int() with base 10: 'oui'
>>> float(y)
ValueError: could not convert string to float: 'oui'
>>> int("0000000")
0
>>> float("111111")
111111
```

```
>>> z = int(input())
```

```
>>> type(z)      <class 'int'>
```

```
>>> if z != 3.0 : print("on arrête là ?")
                  on n'arrête là ?
```

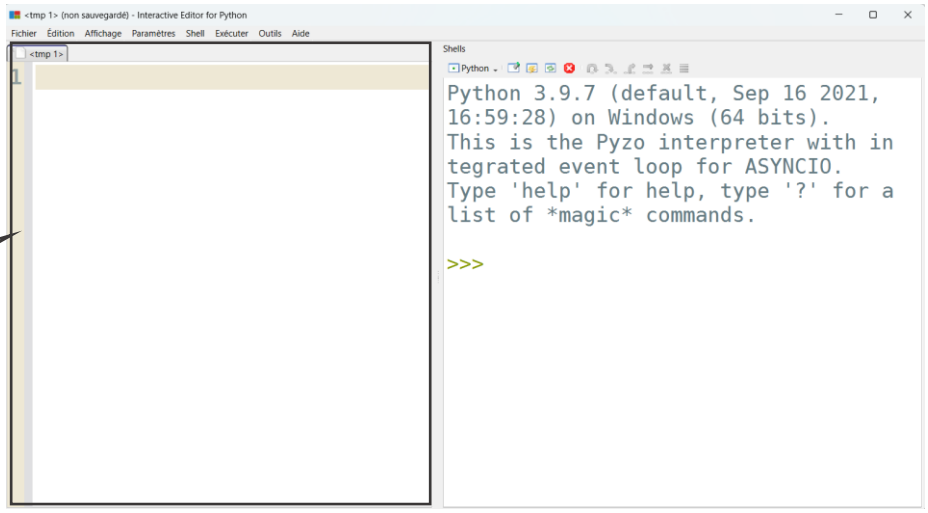
6- LIGNES DE CALCUL ECRITES DANS UN FICHIER

On s'intéresse au programme de calcul suivant :

- Choisir deux nombres positifs dont l'un est le triple de l'autre
- Soustraire 2 au plus petit des deux nombres, ajouter 6 à l'autre, puis calculer le produit des deux nombres obtenus
- Soustraire au résultat, le produit des deux nombres choisis au départ

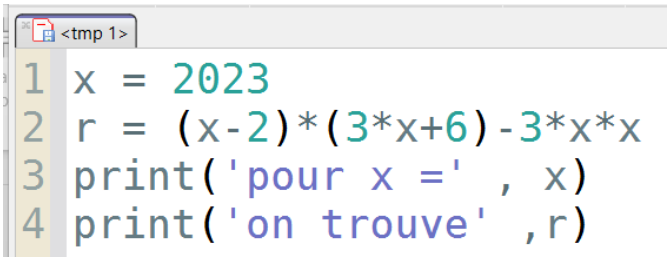
Si le premier nombre choisi est noté x , le second sera $3 \times x$. En suivant le programme de calcul on trouve le résultat suivant : $(x-2) \times (3 \times x + 6) - 3 \times x \times x$

On écrit par contre ici, les lignes de commande, dans l'éditeur sur la fenêtre de gauche de ***Pyzo***. L'avantage c'est que l'on peut écrire plusieurs lignes de commande qui seront exécutées l'une après l'autre, de la première en haut, à la dernière en bas.



Fenêtre nommée *Editeur* qui visualise le contenu d'un fichier qui n'a pour l'instant pas été enregistré (nom : tmp1)

⇨ Ecrire les 4 lignes suivantes dans l'éditeur et exécuter les **en appuyant sur la touche F5 du clavier** …



…. normalement il s'affiche dans la console :

```
>>> (executing file "<tmp 1>")
pour x = 2023
on trouve -12
```

⇨ On peut modifier la valeur de x , normalement on retrouve toujours -12 .
Par exemple :

<tmp 1>

```
1 x = 1000000000
2 r = (x-2)*(3*x+6)-3*x*x
3 print('pour x =' , x)
4 print('on trouve' ,r)
```

⇨ Essayer avec d'autres valeurs de x .

⇨ Prendre un brouillon et réduire mathématiquement l'expression $(x-2)(3x+6)-3\,x^2$ afin de montrer que le résultat issu de ce programme de calcul est bien toujours égal à -12, quel que soit la valeur de la variable x .

On demande **à partir de ce point** de rédiger un compte-rendu au format *.doc* ou *.odt* à transférer en fin d'activité par l'intermédiaire de l'onglet ***Mon Compte*** du site https://nsibranly.fr en utilisant le code : **tp1** . Ce compte-rendu contiendra :

- les réponses aux différentes questions posées,
- les captures d'écran **des morceaux de codes** écrits **et** celles **des résultats des exécutions**. Pour faire ces captures, utiliser *l'Outil Capture d'écran* de Windows.

Repérer correctement les titres de paragraphe : Vous commencerez par exemple ce compte-rendu de la manière suivante :

Info 1 – Bases pour débuter en Python de Martin Dupont – 1G3

6 - Lignes de calcul écrites dans un fichier

$(x-2)\times(3\times x+6)-3\times x\times x =$ (x-2)(3x+6)-3x² = 3x²+6x+…….

Le résultat est bien indépendant de x, donc normal de trouver toujours -12 .

7 - Avec un peu de graphisme c'est mieux

A vous de jouer …..

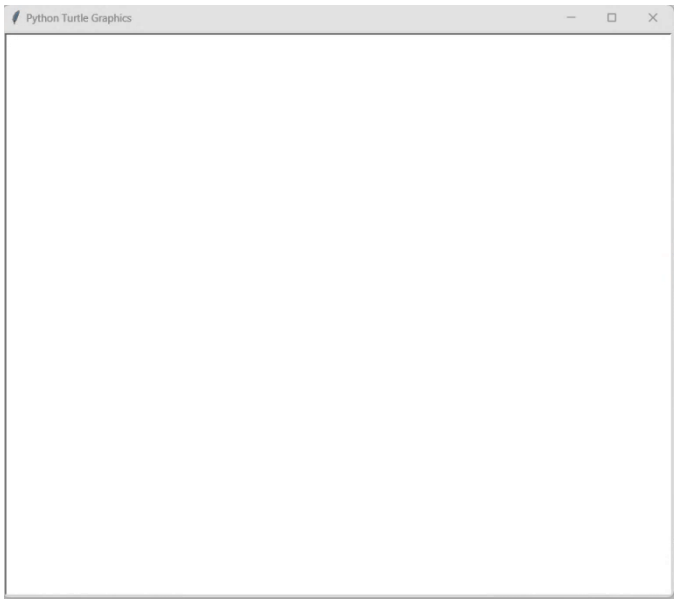
7- AVEC UN PEU DE GRAPHISME C'EST MIEUX :

On utilise ici à nouveau le calcul littéral, mais dans un contexte graphique.

⇨ Effacer les lignes écrites dans l'éditeur précédemment pour les remplacer par les 2 lignes ci-contre :

<tmp 2>

```
1 from turtle import *
2
3
4
5 mainloop()
```



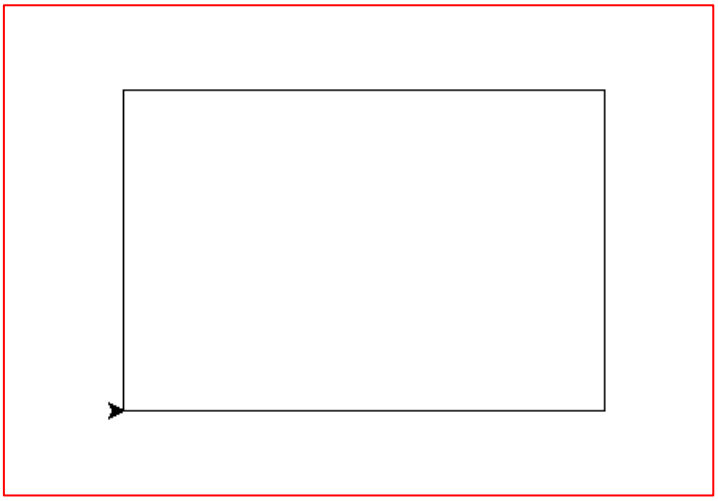
⇨ Exécuter ces 2 lignes (touche F5 du clavier). Elles provoquent ici l'ouverture d'une nouvelle fenêtre graphique, vide pour l'instant ….

⇨ Fermer cette nouvelle fenêtre et exécuter à présent le code ci-dessous :

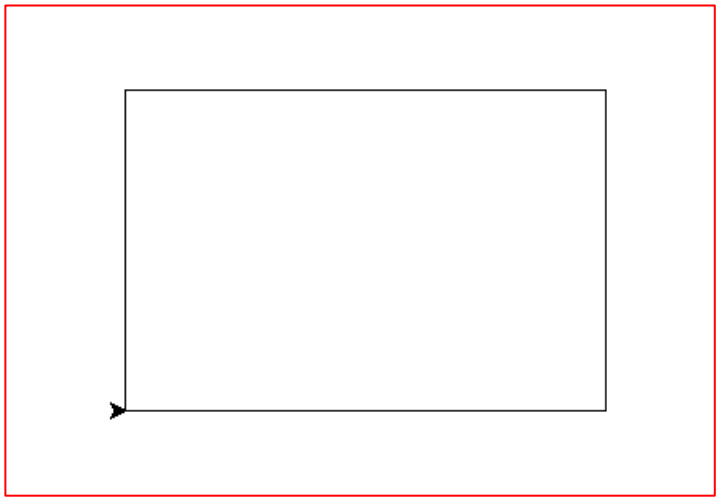
<tmp 2>

```
1  from turtle import *
2
3  forward(300)
4  left(90)
5  forward(200)
6  left(90)
7  forward(300)
8  left(90)
9  forward(200)
10 left(90)
11
12 mainloop()
```

…… normalement vous obtenez le dessin d'un rectangle de 300 pixels de large et 200 pixels de haut.

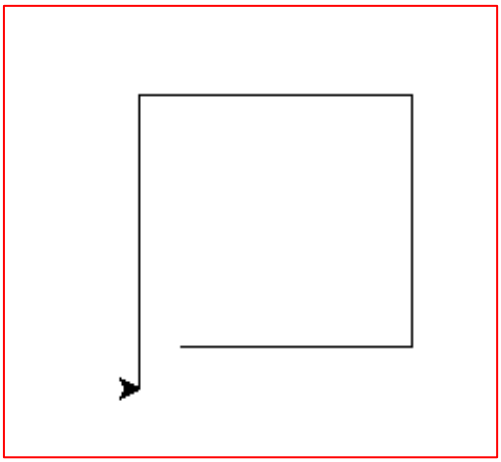


⇨ Exécuter à présent le code ci-contre :



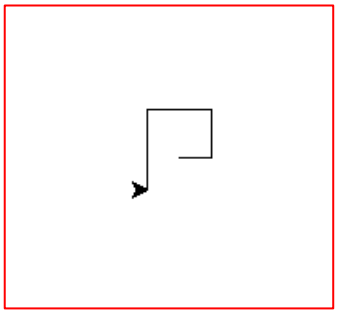
```
<tmp 2>
1  from turtle import *
2
3  a = 300
4  b = 200
5
6  forward(a)
7  left(90)
8  forward(b)
9  left(90)
10 forward(a)
11 left(90)
12 forward(b)
13 left(90)
14
15 mainloop()
```

⇨ Exécuter à présent le code ci-contre :



```
<tmp 2>
1  from turtle import *
2
3  a = 100
4
5  forward(a+10)
6  left(90)
7  forward(a+20)
8  left(90)
9  forward(a+30)
10 left(90)
11 forward(a+40)
12 left(90)
13
14
15 mainloop()
```

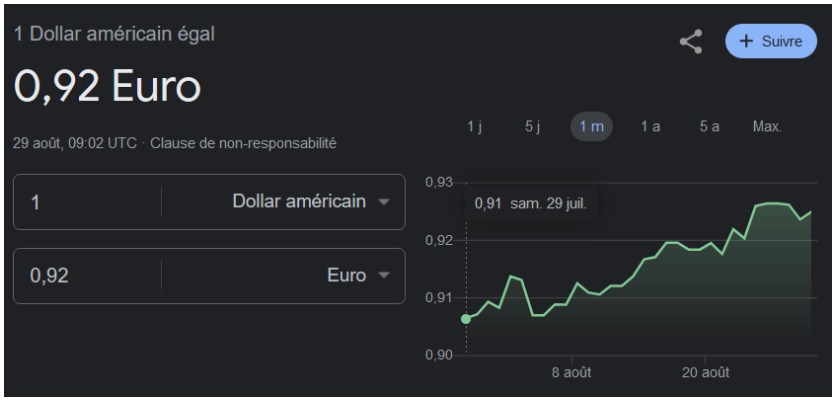
⇨ Et exécuter enfin ce dernier code :



```
<tmp 2>
1  from turtle import *
2
3  a = 10
4
5  forward(2*a)
6  left(90)
7  forward(3*a)
8  left(90)
9  forward(4*a)
10 left(90)
11 forward(5*a)
12 left(90)
13
14
15 mainloop()
```

8- UN PREMIER PETIT CODE SANS GRAPHISME :

Le Dollar américain valait 0,92 € fin août 2023 :



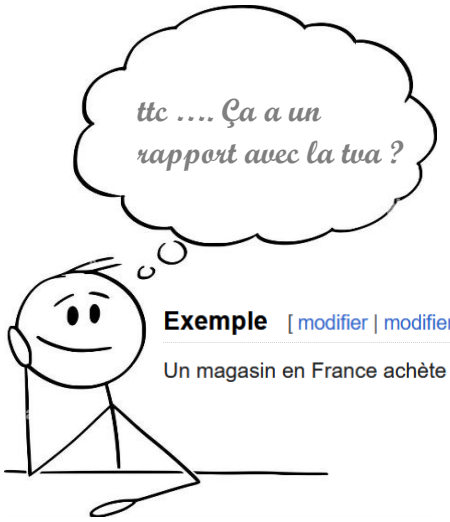
⇨Ecrire dans l'éditeur de pyzo, un code qui saisit une valeur de somme d'argent en € pour ensuite afficher la même somme convertie en $US .

En exécutant votre code vous devrez obtenir dans la console, pour une valeur saisie de 100 :

```
>>> (executing file "conversion_en_dollarsUS.py")
Somme en € : 100
100.0 € font  108.69565217391303 $US
```

```
1  enEuro = float(input("Somme en € : "))
2
3  enDollars = enEuro / 0.92
4
5  #possibilité 1 : dans print(), plusieurs eléments séparés par ,
6  print(enEuro , "€ font " , enDollars , "$US")
7
8  #possibilité 2 : on utilise le f comme format
9  print(f"{enEuro} € font {enDollars} $US")
10
11 #possibilité 3 : on crée un string appelé ici phrase
12 phrase = str(enEuro) + " € font " + str(enDollars) + " $US"
13 print(phrase)
```

Corrigés possibles : les 3 possibilités donnent le même résultat



Sur Wikipédia, on trouve :

La **taxe sur la valeur ajoutée** ou **TVA** est un impôt indirect sur la consommation.

Exemple [modifier | modifier le code]

Un magasin en France achète un stylo 1,00 € HT à son fournisseur, il lui est facturé 1,20 € TTC dont 0,20 € de TVA pour une TVA à 20 %.

⇨ Ecrire un programme qui saisit le prix HT d'un produit pour ensuite en afficher le prix TTC.

En exécutant votre code vous devrez obtenir dans la console, pour une valeur saisie de 10 :

```
>>> (executing file "prix_TTC.py")
Prix HT : 10
Le prix ttc est de 12.0 €
```

Corrigés possibles : les 3 possibilités donnent le même résultat

```
1  prix_ht = float(input("Prix HT : "))
2
3  prix_ttc = prix_ht * 1.2
4
5  #possibilité 1 : dans print(), plusieurs eléments séparés par ,
6  print("Le prix ttc est de",prix_ttc,"€")
7
8  #possibilité 2 : on utilise le f comme format
9  print(f"Le prix ttc est de {prix_ttc} €")
10
11 #possibilité 3 : on crée un string appelé ici message
12 message = "Le prix ttc est de "+str(prix_ttc)+" €"
13 print(message)
```

Remarques :

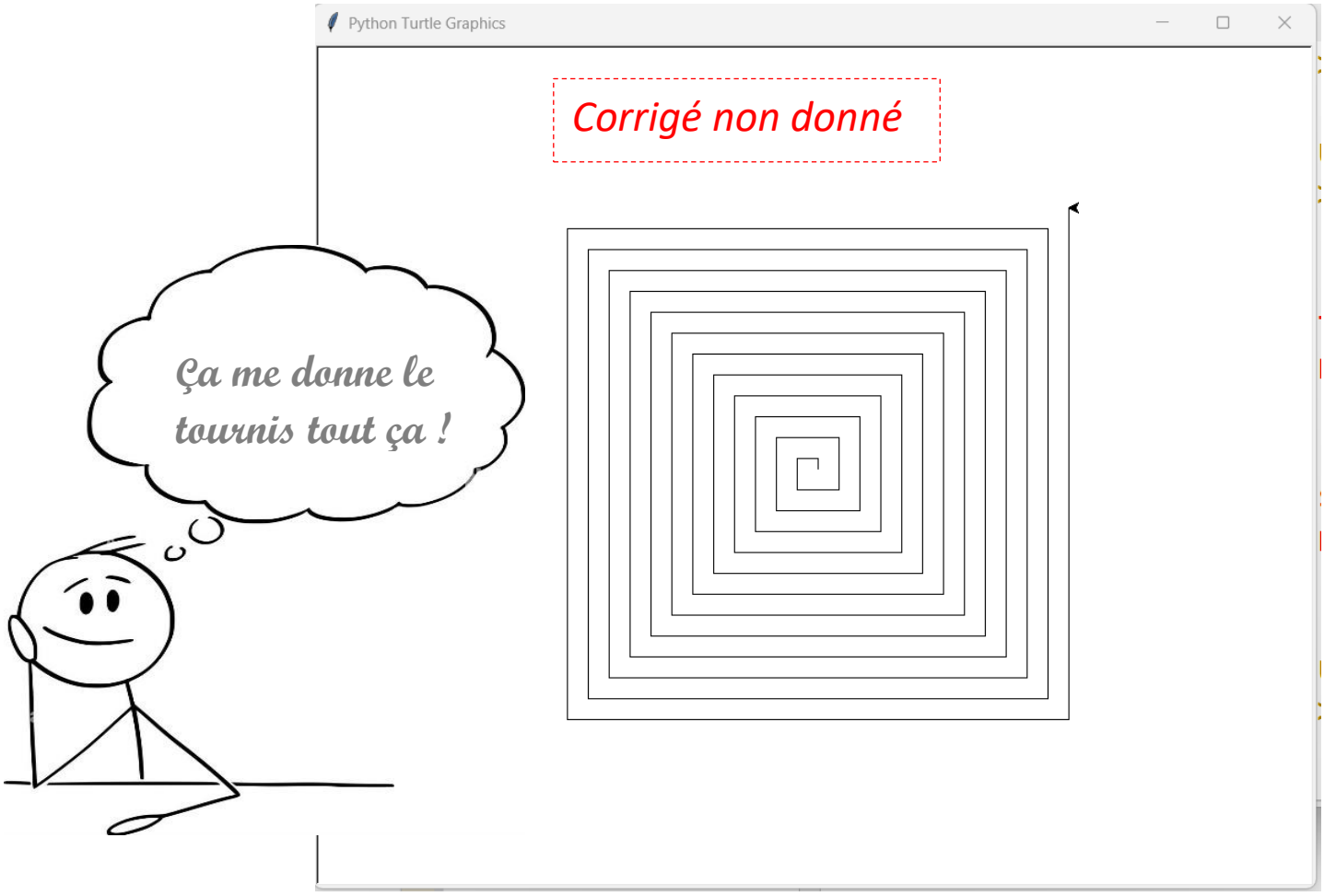
- Dans la possibilité 1 ci-dessus, on a différents arguments dans les parenthèses du print(). Ces arguments peuvent des strings, mais aussi des nombres.

- Dans la possibilité 2, les variables placées entre guillemets peuvent être des strings ou des nombres.

- Dans la possibilité 3, la variable nommée ici *message* est un string obtenu par concaténation de strings (on utilise l'opérateur +). Il est ainsi nécessaire d'utiliser la fonction str() pour convertir les nombres en string.

10- APPLICATION :

⇨ Ecrire un code (ne pas hésiter à faire du copié-collé), qui permet d'obtenir la figure ci-contre :



11- DEUX AUTRES PETITS CODES VRAIMENT TRES UTILES :

⇨ Ecrire un programme qui saisit l'âge de l'utilisateur en années entières (16 par exemple) pour ensuite afficher ce même âge en secondes.
En exécutant votre code vous devrez obtenir dans la console, pour une valeur saisie de 16.5

```
>>> (executing file "age_en_secondes.py")
Entre ton âge : 16.5
Depuis ta naissance, il s'est déja écoulé 520700400.0 secondes
```

```
1 # saisi des valeurs
2 age = float(input("Entre ton âge : "))
3
4 # calcul en secondes
5 age = age * 365.25 * 24 * 60 * 60
6 # affichage du résultat
7 message = "Depuis ta naissance, il s'est déja écoulé " + \
8           str(age) + " secondes"
9 print(message)
```

⇨ Ecrire un programme qui saisit l'âge de l'utilisateur en secondes, pour ensuite afficher ce même âge en années entières et jours additionnels (par exemple : 16 ans et 207 jours).
En exécutant votre code vous devrez obtenir dans la console, pour une valeur saisie de 1 000 000 000

```
>>> (executing file "age_en_annees.py")
Entre ton âge en secondes : 1000000000
Tu as 31.0 ans et 251.32407407407482 jours
```

```
1 age = float(input("Entre ton âge en secondes : "))
2
3 age = age / (60*60*24)  # calcul en jours
4 annee = age // 365.25
5 reste = age % 365.25
6  # affichage
7 message = "Tu as "+str(annee)+" ans et "+str(reste)+" jours"
8 print(message)
```